– Proofs – Software Verification via Three-Valued Abstraction and k-Induction

Nils Timm, Stefan Gruner, and Matthias Harvey

Department of Computer Science, University of Pretoria, Pretoria, South Africa
{ntimm,sgruner}@cs.up.ac.za

Lemma 1. *Let* ${}_{A^r}[\![M,\psi]\!]_k$ *be the enhanced encoding of* ${}_{A^r}[M,S_0 \models_\exists \psi]_k$*. Moreover, let* C *over* DL *be a definite constraint. Then*

$$ {}_{A^r}[\![M,\psi]\!]_k \vdash C \;\Rightarrow\; {}_{A^r}[M,S_0 \models_\forall (\psi \rightarrow btl(C))]_k = true $$

Proof.
The premise is ${}_{A^r}[\![M,\psi]\!]_k^+ \vdash C \vee {}_{A^r}[\![M,\psi]\!]_k^- \vdash C$. Since constraints learned for the over-approximation are also valid constraints of the under-approximation and vice versa, we can conclude that the following holds: ${}_{A^r}[\![M,\psi]\!]_k^+ \models C$ and ${}_{A^r}[\![M,\psi]\!]_k^- \models C$. The definition of the semantic consequence '$\models$' lets us conclude that the following holds: $\forall\,\mathcal{A} : Atoms \rightarrow \{true, false\} : \mathcal{A}({}_{A^r}[\![M,\psi]\!]_k^+) = true \;\Rightarrow\; \mathcal{A}(C) = true$ and $\mathcal{A}({}_{A^r}[\![M,\psi]\!]_k^-) = true \;\Rightarrow\; \mathcal{A}(C) = true$. We now prove by induction on the structure of C that the following holds: $\forall\,\mathcal{A} : Atoms \rightarrow \{true, false\} : \mathcal{A}(C) = true \;\Rightarrow\; \mathcal{A}([\![btl(C)]\!]_k^+) = true$ and $\mathcal{A}([\![btl(C)]\!]_k^- = true$:

1. Let $C = p[t]_i$. The premise is $\mathcal{A}(p[t]_i) = true$. We have that $btl(p[t]_i) = p_i$, $[\![btl(p[t]_i)]\!]_k^+ \equiv p[u]_i \vee p[t]_i$, and $[\![btl(p[t]_i)]\!]_k^- \equiv p[t]_i$. From the premise we immediately get $\mathcal{A}([\![btl(p[t]_i)]\!]_k^+) = true$ and $\mathcal{A}([\![btl(p[t]_i)]\!]_k^-) = true$.
2. Let $C = p[f]_i$. The premise is $\mathcal{A}(p[f]_i) = true$. We have that $btl(p[f]_i) = \neg p_i$, $[\![btl(p[f]_i)]\!]_k^+ \equiv p[u]_i \vee p[f]_i$, and $[\![btl(p[f]_i)]\!]_k^- \equiv p[f]_i$. From the premise we immediately get $\mathcal{A}([\![btl(p[f]_i)]\!]_k^+) = true$ and $\mathcal{A}([\![btl(p[f]_i)]\!]_k^-) = true$.
3. Let $C = l_j[d]_i$. The premise is $\mathcal{A}(l_j[d]_i) = true$. We have that $btl(l_j[d]_i) = \bigvee_{(l_m \ldots l_0) \in Loc_j, l_d = 1} (pc_j = l_m \ldots l_0)_i$, $[\![btl(l_j[d]_i)]\!]_k^+ = \bigvee_{(l_m \ldots l_0) \in Loc_j, l_d = 1} \bigwedge_{d'=0}^{m} (\text{if } l_{d'} = 1 \text{ then } l_j[d']_i \text{ else } \neg l_j[d']_i) \equiv l_j[d]_i$, and $[\![btl(l_j[d]_i)]\!]_k^- \bigvee_{(l_m \ldots l_0) \in Loc_j, l_d = 1} \bigwedge_{d'=0}^{m} (\text{if } l_{d'} = 1 \text{ then } l_j[d']_i \text{ else } \neg l_j[d']_i) \equiv l_j[d]_i$. From the premise we immediately get $\mathcal{A}([\![btl(l_j[d]_i)]\!]_k^+) = true$ and $\mathcal{A}([\![btl(l_j[d]_i)]\!]_k^-) = true$.
4. Let $C = \neg l_j[d]_i$. The premise is $\mathcal{A}(\neg l_j[d]_i) = true$. We have that $btl(\neg l_j[d]_i) = \bigvee_{(l_m \ldots l_0) \in Loc_j, l_d = 0} (pc_j = l_m \ldots l_0)_i$, $[\![btl(\neg l_j[d]_i)]\!]_k^+ = \bigvee_{(l_m \ldots l_0) \in Loc_j, l_d = 0} \bigwedge_{d'=0}^{m} (\text{if } l_{d'} = 1 \text{ then } l_j[d']_i \text{ else } \neg l_j[d']_i) \equiv \neg l_j[d]_i$, and $[\![btl(\neg l_j[d]_i)]\!]_k^- \bigvee_{(l_m \ldots l_0) \in Loc_j, l_d = 0} \bigwedge_{d'=0}^{m} (\text{if } l_{d'} = 1 \text{ then } l_j[d']_i \text{ else } \neg l_j[d']_i) \equiv \neg l_j[d]_i$. From the premise we immediately get $\mathcal{A}([\![btl(\neg l_j[d]_i)]\!]_k^+) = true$ and $\mathcal{A}([\![btl(\neg l_j[d]_i)]\!]_k^-) = true$.
5. Let $C = c_1 \vee \ldots \vee c_m$. The premise is $\mathcal{A}(c_1 \vee \ldots \vee c_m) = true$. Hence, $\mathcal{A}(c_1) = true \vee \ldots \vee \mathcal{A}(c_m) = true$. Since for each literal $c_1, \ldots, c_m$ one of the cases 1 to 4 must apply, we immediately get $\mathcal{A}([\![btl(c_1 \vee \ldots \vee c_m)]\!]_k^+) = true$ and $\mathcal{A}([\![btl(c_1 \vee \ldots \vee c_m)]\!]_k^-) = true$.

By combining what we have proven so far we get $\forall\,\mathcal{A} : Atoms \rightarrow \{true, false\} : \mathcal{A}({}_{A^r}[\![M,\psi]\!]_k^+) = true \;\Rightarrow\; \mathcal{A}([\![btl(C)]\!]_k^+) = true$ and $\mathcal{A}({}_{A^r}[\![M,\psi]\!]_k^-) = true \;\Rightarrow\; \mathcal{A}([\![btl(C)]\!]_k^-) = true$. Since we have a one-to-one correspondence between assignments to an encoded bounded model checking problem and paths of the actual bounded model checking problem [1], we can conclude the following: $\forall\,\pi$ of $M : [\pi \models \psi]_k \Rightarrow [\pi \models btl(C)]_k$ holds. This is equivalent to $[M, S_0 \models_\forall (\psi \rightarrow btl(C))] = true$ which completes the proof of Lemma 1. $\square$

Lemma 2. *Let* ${}_{A^r}[\![M,\psi]\!]_k$ *be the encoding of* ${}_{A^r}[M, S_0 \models_\exists \psi]_k$ *and let* C *be a definite constraint. Then*

$$ {}_{A^r}[M, S_0 \models_\forall (\psi \rightarrow btl(C))]_k = true \;\Rightarrow\; {}_{A^r}[\![M,\psi]\!]_k \models C $$

Proof.
The premise is that ${}_{A^r}[M, S_0 \models_\forall (\psi \rightarrow btl(C))]_k = true$ holds. Corollary 1 and Theorem 2 together allow us to transfer this definite model checking result to the refined model checking problem, i.e. ${}_{A^{r'}}[M, S_0 \models_\forall (\psi \rightarrow btl(C))]_k = true$ holds as well. Hence, we have that the BTL property $\psi \rightarrow btl(C)$ holds universally for the Kripke structure M over $A^{r'}$. Consequently, the following equivalence holds: ${}_{A^{r'}}[M, S_0 \models_\exists \psi]_k \equiv {}_{A^{r'}}[M, S_0 \models_\exists \psi \wedge (\psi \rightarrow btl(C))]_k$, which is equivalent to ${}_{A^{r'}}[M, S_0 \models_\exists \psi \wedge btl(C)]_k$. From Definition 4 ($sat_3$) we immediately get $sat_3({}_{A^{r'}}[\![M,\psi]\!]_k) \equiv sat_3({}_{A^{r'}}[\![M,\psi]\!]_k \wedge C)$ which completes the proof of Lemma 2. □

References

1. Timm, N., Gruner, S., Harvey, M.: A bounded model checker for three-valued abstractions of concurrent software systems. In: Ribeiro, L., Lecomte, T. (eds.) Formal Methods: Foundations and Applications. pp. 199–216. Springer (2016)